FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 40, No 1 (2025), 13-31
https://doi.org/10.22190/FUMI221113002S
Original Scientific Paper

EMBEDDING AND WEIGHTING OF WEBSITE FEATURES FOR PHISHING DETECTION

Nikola Stevanović

University of Niš, Faculty of Sciences and Mathematics
Department of Computer Science, Višegradska 33, 18000 Niš, Serbia

ORCID ID: Nikola M. Stevanović https://orcid.org/0000-0003-2083-8313

Abstract. One of the most common cyber threats are phishing attacks. During a phishing attack, attackers use various technical and social engineering tricks to try to lure victims to a phishing website. The website looks like it belongs to a trusted organization but is actually run by the attackers and used to mislead victims into revealing their passwords, credit card numbers, or other confidential information. In this paper, we use discrete descriptive website features to detect whether a website is phishing or legitimate. We create a customized embedding layer specifically designed for these types of features, as well as an embedding weighting mechanism that we later apply. We propose a convolutional neural network-based model for phishing website detection and demonstrate its efficacy on three datasets. With accuracy rates of up to 97.56%, the model performed on par with or better than the current state-of-the-art approaches on each dataset.
Keywords: phishing website detection, web attacks, embeddings, weighting of embeddings, cybersecurity, deep learning.

1. Introduction

Nowadays, online services have become an irreplaceable part of our everyday lives. We use them for entertainment, to order some goods, to chat with family and friends, and for work purposes. While surfing the Internet, we are constantly

Received: November 13, 2022, revised: August 15, 2024, accepted: November 22, 2024
Communicated by Marko Petković
Corresponding Author: N. Stevanović
E-mail addresses: nikola.stevanovic@pmf.edu.rs (N. Stevanović)
2020 *Mathematics Subject Classification.* Primary 68T99; Secondary 68T07, 68T30

exposed to cybersecurity threats. One of the most frequent threats in contemporary cybersecurity are phishing attacks. In a phishing attack, an attacker impersonates a trusted organization or person, with the aim of obtaining some private or confidential information. It can be anything which can later be exploited by the attackers to generate some profit, including passwords, bank card numbers and similar.

The attackers usually start by disseminating links to their phishing websites. These links can be spread via electronic mails (emails), as well as through social media sites, forums, and similar platforms. Once the victim clicks on a malicious link, they are redirected to a phishing website. Since the attackers impersonate legitimate organizations, they aim to appear as credible as possible in their communications with victims. In some types of phishing attacks, such as tabnabbing, the victim may even be redirected to a phishing website while visiting the legitimate website that the attackers are impersonating. On these phishing websites, the attackers aim for victims to enter their private information, while believing that they are interacting with the genuine websites of trusted organizations.

According to the Anti-Phishing Working Group [3], the number of phishing attacks doubled over the course of 2020. In January 2021, they recorded an all-time high of 245,771 new phishing sites in that month alone. Additionally, they also documented more than 200,000 attacks in March, which was the fourth-worst month in their reporting history. The same report also states that around 83% of phishing websites use TLS certificates. Nowadays, Internet users are highly vulnerable to phishing attacks, and having a reliable method to detect phishing websites can significantly mitigate the associated risks.

There are many approaches to defending against phishing. One of the most obvious is legal solutions. By creating laws against phishing activities and prosecuting attackers, lawmakers can reduce the number of attacks. However, phishing websites often exist for only a short period, necessitating prompt action by law enforcement authorities. Another approach is educating Internet users about cyber threats. Although this method can be effective, it is usually difficult to implement, because it requires users to spend considerable time learning about phishing methods. Additionally, attackers are becoming increasingly sophisticated in their techniques for mimicking legitimate websites.

The most common approach to defending against phishing attacks are technical solutions. Some methods detect sources that spread links to phishing websites, such as phishing emails [13, 21]. Other methods attempt to detect phishing websites directly, and the approach that we use falls into this category. Earlier methods of phishing website detection relied on blacklists. By maintaining a list of known phishing URLs, we can easily check if a requested URL is on that list. However, this approach suffers from a high number of false negatives, as it takes time for a phishing website to be detected and added to the list. This delay can provide attackers with sufficient time to achieve their goals.

Machine learning models are used to make defense mechanisms better at detecting previously unseen phishing websites. By extracting useful features from phishing websites and applying machine learning algorithms, it is possible to iden-

tify a significantly larger number of phishing websites. Discrete descriptive features of websites can aid in the better interpretation of models and provide a clearer understanding of the factors that influence classification results the most.

In this paper, we propose a convolutional neural network-based approach for phishing website detection. Drawing inspiration from the use of embeddings in natural language processing [15], we first designed a website feature embedding layer. This layer is tailored to the discrete descriptive website features commonly used in the literature. Next, we created an adaptation of the CancelOut [9] weighting layer for use with website feature embeddings. Finally, we designed a convolutional neural network architecture that utilizes these weighted embeddings. To demonstrate the effectiveness of our approach, we conducted extensive testing and analysis on three datasets, including the commonly used Phishing Websites Dataset. Experimental results indicate that our approach performs as well as or better than the best existing approaches.

The remainder of the paper is organized as follows. In Section 2. we provide an overview of the current literature on phishing website detection. Section 3. describes our proposed architecture for phishing website detection, including the website embedding and weighting layers that we use. This section also details the datasets used for evaluation. A detailed explanation of the evaluation procedure, implementation details, and evaluation results is provided in Section 4.. Finally, Section 5. presents some concluding remarks and potential ideas for further improvements.

2. Related work

As we can see from the previous section, there is a wide range of approaches to phishing detection. Since our approach relies on detecting phishing websites based on discrete descriptive features, this section reviews similar methods from the literature.

Ali and Ahmed [6] proposed a deep neural network architecture for phishing website detection. By utilizing genetic algorithm-based feature selection and weighting, they achieved accuracy rates of 90.39% and 91.13%, respectively. Ali and Malebary [7] applied particle swarm optimization for phishing website feature weighting. They evaluated their approach using a 10-fold cross-validation technique and obtained the highest accuracy of 96.83% with a random forest classifier. After experimenting with three different feature selection algorithms (correlation-based feature subset, information gain and Chi-Square), Thabtah and Abdelhamid [28] developed a PART-based classifier using only two input features. The classifier achieved an accuracy rate of 91.26% in 10-fold cross-validation.

Mohammad et al. [20] proposed a self-structuring neural network approach with one hidden layer for phishing website detection. The algorithm is able to automatically adjust the learning rate and incrementally adds new neurons to the hidden layer. Their approach achieved an accuracy rate of 92.48% on the testing dataset. Hadi et al. [12] approached the problem with an associative classification

algorithm that employs association rules to make predictions. This model is easy to interpret and achieved an accuracy rate between 92% and 93%. Another associative classification approach was proposed by Alqahtani [8], which attained an accuracy rate of 95.20% and an F_1-score of 95.11%.

Rajab [27] utilized two feature selection techniques and the data mining algorithm RIPPER. When selecting 11 and 9 features, the classification error was only 1.32% and 1.02% higher than that with all 30 features, respectively. Penmatsa and Kakarlapudi [26] proposed a rough set-based ant colony optimization technique and selected 23 features from the original dataset. After applying a random forest classifier, they achieved an accuracy rate of 97.26% and an F_1-score of 97.3%. To detect phishing websites, Motlagh and Bardsiri [22] employed a learning-based optimization algorithm to adjust the weights of a multilayer perceptron with two hidden layers. Their approach attained an accuracy rate of 93.42%. Abad et al. [1] applied three machine learning models (artificial neural network, support vector machine and adaptive boosting) to detect phishing websites. They obtained the best results with the support vector machine model, which achieved a testing accuracy of 96.71%.

Al-Ahmadi and Lasloum [2] proposed a multilayer perceptron-based model and obtained an accuracy rate of 96.65% and an F_1-score of 96.65% when classifying between phishing and legitimate websites. Vrbančič et al. [29] utilized two swarm intelligence algorithms, the bat [31] and hybrid bat [10] algorithms, to adjust the hyperparameters of a neural network (the number of epochs, batch size, learning rate and the number of neurons in the first hidden layer). The best model obtained, which uses the bat algorithm to determine hyperparameters, achieved an accuracy rate of 96.5% in a 10-fold cross-validation evaluation. The same authors later also applied the firefly swarm intelligence algorithm [30], and obtained an accuracy rate of 96.65% and an F_1-score of 96.61%.

Thabtah et al. [35] developed a neural network solution for phishing detection. Their algorithm dynamically tunes the structural parameters of the network during the training phase, with the aim of obtaining a non-overfitting classifier with high accuracy. Their model attained an accuracy rate of 93.06%. Al-Sarem et al. [4] proposed a multi-step method for phishing website detection. They first trained several machine learning models without hyperparameter optimization. Subsequently, they improved the models by using a genetic algorithm to find optimal hyperparameters. The best models were then selected to create a stacking ensemble method, which achieved an accuracy rate of 97.16%. Jalal et al. [39] experimented with random forest, decision tree, linear model and neural network algorithms. They achieved the highest accuracy rate of 95.7% with the random forest classifier.

Lakshmi et al. [5] developed a deep learning approach for phishing website detection using the Adam optimizer, and achieved an accuracy of 96%. Parra et al. [36] proposed a distributed deep learning framework for phishing attack detection. A convolutional neural network was used to detect phishing websites, and it achieved an accuracy rate of 94.3%. Al-Milli et al. [37] proposed a 1-dimensional convolutional neural network for phishing website detection. They obtained an

accuracy rate of 94.31% and an AUC rate of 91.23%. A binary LSTM neural network classifier was proposed by Wang et al. [38]. It outperformed a random forest classifier and achieved an accuracy rate of 95.47%. As a deep learning approach, our solution is the most similar to the methods presented in [2, 5, 6, 35, 36, 37, 38].

3. Proposed approach

3.1. Datasets

To create an effective machine learning-based phishing website detection model, it is essential to select appropriate features from websites. According to Mohammad et al. [19], there are four categories of relevant features for phishing website detection: address bar-based features, abnormality-based features, HTML and JavaScript-based features and domain-based features. All of these features can be automatically extracted from websites without relying on human expertise in the extraction process. In addition to being useful for detecting phishing websites, these features are also descriptive, which enhances the interpretability of model decisions. Due to all these advantages, we have decided to use these features in our detection model.

To evaluate our approach, we have used a total of three datasets. The first dataset used (Dataset-1) is the Phishing Websites Dataset, which is frequently used in phishing website detection literature. This dataset contains 11,055 websites, of which 4,898 are phishing websites and 6,157 are legitimate websites. Each website is represented by 30 features: 12 address bar-based features, 6 abnormality-based features, 5 HTML and JavaScript-based features and 7 domain-based features. All the features are listed in Table 3.1.

The second dataset (Dataset-2) contains 2,456 samples. Each website is represented by the same 30 features as in Dataset-1. It contains 1,362 phishing websites and 1,094 legitimate websites. Both Dataset-1 and Dataset-2 are stored on the UCI Machine Learning Repository [16], where a detailed description of the features can be found.

The third dataset (Dataset-3) was created by Neda Abdelhamid and is stored in the University of Irvine machine learning repository [17]. Each website in this dataset is represented by 9 features. It contains 702 phishing and 548 legitimate samples. The dataset also contains 103 samples that are not labeled as either phishing or legitimate, and those samples are excluded from the analysis. A detailed description of the dataset features is provided in the introductory paper [18].

Each feature has two or three possible values. It can have a value of -1, which indicates that based on that feature the website is more likely to be phishing, or 1, when the feature indicates that the website is more likely to be legitimate. A feature with three possible values can also have a value of 0, indicating that the website could be suspicious.

Table 3.1: Phishing Websites Dataset Features.

Address bar-based features
Using the IP Address
Long URL to Hide the Suspicious Part
Using URL Shortening Services “TinyURL”
URL’s having “@” Symbol
Redirecting using “//”
Adding Prefix or Suffix Separated by (-) to the Domain
Subdomain and Multi Subdomains
HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)
Domain Registration Length
Favicon
Using Non-Standard Port
The Existence of “HTTPS” Token in the Domain Part of the URL
Abnormality-based features
Request URL
URL of Anchor
Links in <Meta>, <Script> and <Link> tags
Server Form Handler (SFH)
Submitting Information to Email
Abnormal URL
HTML ad JavaScript-based features
Website Forwarding
Status Bar Customization
Disabling Right Click
Using Pop-up Window
IFrame Redirection
Domain-based features
Age of Domain
DNS Record
Website Traffic
PageRank
Google Index
Number of Links Pointing to Page
Statistical-Reports-Based Feature

An example of a feature that has two possible values is ”Using the IP Address”. Specifically, if an IP address is used instead of a domain name (for example, ”http://125.98.3.123/fake.html”), the feature value will be -1, indicating that the website is more likely to be phishing. If a domain name is present in the URL, the feature value will be 1. An example of a feature with three possible values is

"Subdomain and Multi Subdomains". If the URL does not contain subdomains, the value is 1, if it contains one subdomain, the value is 0, and if it contains two or more subdomains, the value is -1. Attackers sometimes use multiple subdomains to hide the original domain name of their website and mislead victims.

3.2. Website Feature Embedding

Since values -1 and 1 indicate phishing and legitimate characteristics of a feature, respectively, and value 0 represents something between these two characteristics, we wanted to incorporate this into our website feature embedding. To the best of our knowledge, we are the first to use embeddings of discrete descriptive phishing website features.

Each input feature is represented by two embedding vectors: one for phishing characteristics (e^i_{-1}) and one for legitimate characteristics (e^i_1). Both of them are d-dimensional vectors of real numbers, where d represents the embedding size. In our case, d is set to 100. The superscript term i indicates the index of the feature. If we denote with n the number of input features, there is a total of $2n$ embedding vectors.

To embed the input features of a website, we do the following. If the value of the i-th website feature is -1, its embedding is e^i_{-1}. If the value is 1, the embedding of the feature is e^i_1. When the feature value is 0, we want to combine the embedding vectors for legitimate and phishing characteristics, so we model the embedding of the feature with $(e^i_{-1} + e^i_1)/2$ in this case.

This embedding layer transforms a website feature vector into a matrix of size $n \times d$. We later apply weighting on that matrix, and then it is used as the input of a convolutional model, as explained in the next two sections. The parameters of all embedding vectors are learnable and are adjusted during the training phase, together with all other parameters of the model.

3.3. Weighting of Website Feature Embeddings

Since it is difficult to understand what influences the decisions of deep learning models, they are often described as black-box models. This is usually not the case with traditional machine learning models. In linear models, each feature is associated with a single weight that describes its influence on the decision. In decision tree-based models, each node can be easily interpreted. One strategy that helps in the interpretation of deep learning models is feature weighting.

The use of feature weighting and feature selection for better interpretability has already been explored in the phishing website detection literature [6, 7, 26, 28]. In current approaches, it is done as a two-step process. The first step is feature weighting, and then in the next step a model is trained using the features with the highest weights. Conducting these tasks independently reduces the potential for model training and feature weighting to mutually enhance each other.

End-to-end approaches in deep learning are gaining in popularity in recent years. They simplify the process of creating, storing and using deep learning models, and typically achieve better results. One method that facilitates feature weighting and neural network model training in an end-to-end approach is presented in the CancelOut [9] paper.

In their approach, a model receives N-dimensional input vectors. The weighting layer contains a vector of weights $W_{CO} \in \mathbb{R}^N$. They use sigmoid nonlinearity to bound the weights within the range of 0 to 1. If we denote the input vector as x and the sigmoid function (3.1) as σ, the output of the layer is the elementwise multiplication between the input values and bounded weights, $x \odot \sigma(W_{CO})$.

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{3.1}$$

In our case, we do not weight individual input values. Instead, we aim to weight the input website features, which are represented by their embedding vectors. Let us label with $X \in \mathbb{R}^{n \times d}$ the output of the embedding layer for a single website. It contains n embedding vectors, one per each input feature. We first normalize each of them using the Euclidean norm. Our weighting layer has a vector of n parameters, $W_{CO} \in \mathbb{R}^n$. Instead of applying weighting on individual values, each weighting parameter is responsible for one input feature, which is represented by a single row in X. We also use sigmoid nonlinearity to bound weighting parameters, and the output of our weighting layer is given in the Equation (3.2).

$$\mathrm{diag}(\sigma(W_{CO}))\,\mathrm{row\_norm}(X) \tag{3.2}$$

We denote with $\mathrm{diag}(\sigma(W_{CO}))$ diagonal matrix constructed from the elements of the vector $\sigma(W_{CO})$, and with $\mathrm{row\_norm}(X)$ the matrix obtained by normalizing each row of the matrix X using the Euclidean norm. The product of these two matrices, $\mathrm{diag}(\sigma(W_{CO}))$ and $\mathrm{row\_norm}(X)$, is the standard matrix multiplication. The output of this layer can then be used by the rest of the network, and all the parameters can be adjusted in an end-to-end training process.

3.4. Architecture

After embedding the original website features, and applying weighting to these embeddings, we use a convolutional neural network to combine the embeddings and create more complex features. The architecture contains two convolutional layers. Each of them has 100 kernels. The kernel size is 13 for the models trained on Dataset-1 and Dataset-2, and 3 for the model trained on Dataset-3, since it has a smaller input size.

To achieve more complex nonlinear transformations of the input feature vector, we apply ReLU nonlinearity after each convolutional layer. The nonlinearity is applied elementwise, and its formula is provided in Equation (3.3).

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{3.3}$$

The output of the ReLU nonlinearity after the second convolutional layer is flattened into a vector (of size 600 for Dataset-1 and Dataset-2, and 500 for Dataset-3). We then apply a final linear layer with a single output neuron to this vector. The sigmoid activation of this output neuron represents the probability that the website is phishing.

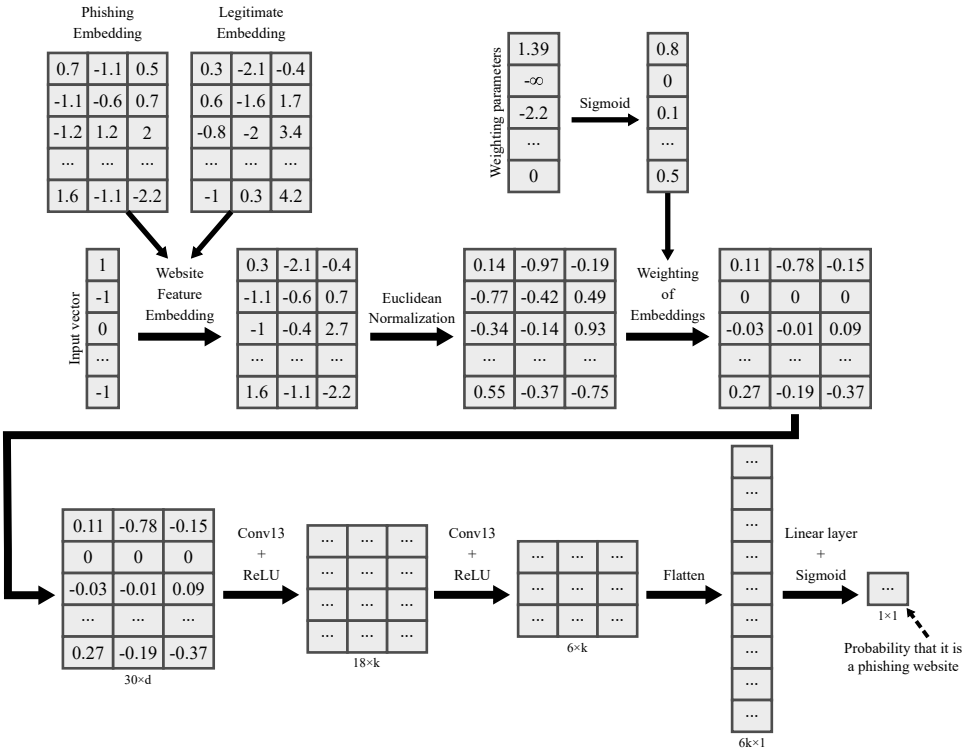

FIG. 3.1: Our proposed architecture.

Our full architecture is depicted in Figure 3.1. The only differences between the figure and the actual architecture are that the embedding size d is 100 instead of 3, as shown in the figure, and the number of kernels k in the convolutional layers is also 100 instead of 3. The size of 3 was used in the figure for demonstration purposes, as it is a smaller number that simplifies the illustration.

3.5. Alternative architectures

Our model uses convolutional layers to create higher-level features and aggregate website feature embeddings. However, this is not the only method for achieving this goal. Another possible strategy is to use recurrent neural network (RNN) cells. To

explore these alternatives and compare them with our original architecture, we have experimented with two alternative models.

The first alternative employs a bidirectional long short-term memory (LSTM) [23] cell for aggregation. This cell takes website feature embeddings as input. The outputs of the last hidden states in each direction are concatenated, and a final linear layer with a single output neuron is applied to this vector. The sigmoid activation of this neuron represents the probability that the website is phishing.

The second alternative is similar to the first, but uses a bidirectional gated recurrent unit (GRU) [24] cell for aggregation instead of the LSTM cell. GRU is a more recent architecture with fewer gating mechanisms than LSTM. It also usually runs faster.

Since both LSTM and GRU use gating mechanisms, it is difficult to justify the use of weighting over embeddings in this case, as this layer functions similarly to their gating mechanisms. Empirical tests in the next section will also confirm this conclusion.

4. Evaluation

To implement our model, we utilized the PyTorch [25] library for machine learning. All experiments were carried out on Google Colab [11], using a graphics processing unit (GPU).

We use binary cross-entropy loss function to train our model. We regularize all model parameters except for the bias parameters. Let us label with W_r a vector that contains all model parameters except the bias and weighting parameters. We add to the binary cross-entropy loss function three regularization terms. The first one is the L_2 regularization of W_r, which aids in mitigating model overfitting. The second and third terms are responsible for the regularization of the weighting parameters (W_{CO}). In order to encourage smaller and diverse weights, we simultaneously maximize their variance and minimize the L_1 norm of their sigmoid activations, similarly to how it was done in the CancelOut layer [9]. The total regularization term is given in Equation (4.1).

$$\frac{\lambda_1}{2} \|W_r\|_2^2 + \lambda_2 \|\sigma(W_{CO})\|_1 - \lambda_3 \mathrm{Var}(W_{CO}) \tag{4.1}$$

We apply Bessel's correction to calculate the variance. To balance the three regularization terms, we use hyperparameters. The exact values used are $\lambda_1 = 10^{-4}$, $\lambda_2 = 2 \cdot 10^{-3}$ and $\lambda_3 = 3 \cdot 10^{-3}$ for Dataset-1, $\lambda_1 = 10^{-4}$, $\lambda_2 = 3 \cdot 10^{-2}$ and $\lambda_3 = 4 \cdot 10^{-2}$ for Dataset-2 and $\lambda_1 = 3 \cdot 10^{-3}$, $\lambda_2 = 2 \cdot 10^{-2}$ and $\lambda_3 = 3 \cdot 10^{-2}$ for Dataset-3. When training the model without weighting of embeddings, we only use the first regularization term from Equation (4.1).

All the parameters of our model are updated by the Adam [14] optimizer, with a starting learning rate of 0.001. Whenever there are more than 5 consecutive epochs without a reduction in training loss, we divide the learning rate by 2. We train

the model for 400 epochs, using a mini-batch size of 200. Before each epoch, we randomly shuffle the order of websites in the dataset and select consecutive websites to form mini-batches.

Since the sizes of the datasets are relatively small, the commonly used train-validation-test split could produce unstable results. Therefore, we decided to use 10-fold cross-validation for evaluation. We first randomly divided all websites into 10 groups of approximately equal size (a group can have at most one website more than any other group). In each of the 10 experiments, websites from one group are used for testing, while the model is trained with websites from the remaining 9 groups. As a result, each website is used exactly once for testing, and these results are used to calculate all evaluation metrics.

We have applied the following evaluation metrics to analyze the performance of our proposed phishing website detection model: accuracy, precision, recall (true positive rate), F_1-score and false positive rate (FPR). Their equations are given below (4.2) in terms of the number of true positives (tp), false negatives (fn), false positives (fp) and true negatives (tn).

$$\begin{aligned} \text{Accuracy} &= \frac{tp+tn}{tp+fn+fp+tn} \\ \text{Precision} &= \frac{tp}{tp+fp} \\ \text{Recall} &= \frac{tp}{tp+fn} \\ \text{F}_1\text{-score} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \\ \text{FPR} &= \frac{fp}{fp+tn} \end{aligned} \tag{4.2}$$

Table 4.1 depicts the classification results of our model, both with and without using weighting of embeddings. In addition to providing a better interpretation of the importance of individual features, the table shows that the model with weighting achieved slightly better results on all three datasets. This model obtained an accuracy rate of 97.53% and an F_1-score of 97.20%, while the model without weighting achieved an accuracy rate of 97.39% and an F_1-score of 97.03% on Dataset-1. Both variants obtained good results, which can be attributed to the strength of the embeddings and convolutional layers. The results are similar on Dataset-2, and slightly lower on Dataset-3, which is a smaller but more challenging dataset.

Classification results for the two alternative approaches are shown in Table 4.2. As previously assumed, both approaches achieved higher accuracies with the variant that did not use weighting on each dataset. The performance of both variants was similar. The LSTM variant achieved slightly higher accuracy on Dataset-1, their accuracies were the same on Dataset-2, and on Dataset-3, the GRU model had slightly higher accuracy. Compared to our proposed model, which uses convolution, both alternatives achieved lower results on each dataset. The difference

Table 4.1: Results of our proposed model.

Model	Accuracy	Precision	Recall	F_1-score	FPR
Dataset-1					
With weighting	0.97531	0.97749	0.96652	0.97197	0.01770
Without weighting	0.97386	0.97663	0.96407	0.97031	0.01835
Dataset-2					
With weighting	0.97557	0.97588	0.98018	0.97802	0.03016
Without weighting	0.9715	0.9757	0.97283	0.97426	0.03016
Dataset-3					
With weighting	0.9336	0.94405	0.93732	0.94067	0.07117
Without weighting	0.9280	0.94348	0.92735	0.93534	0.07117

is most noticeable on Dataset-2, where our proposed model reduced the number of misclassified samples by 37.5%.

In Table 4.3, we compare our model with other approaches in the phishing website detection literature. We can see that our approach is among the best currently available methods. Website feature embeddings and convolutional layers added complexity to our model compared to traditional fully connected feed-forward neural networks [20, 22, 29, 30], resulting in higher accuracy rates. By incorporating a weighting layer into our architecture, we further improved accuracy. Feature selection and weighting is a standard technique frequently used in phishing detection literature [6, 7, 26]. Unlike some other common weighting techniques, our approach simultaneously learns weights and trains the model. This end-to-end approach facilitates a stronger connection between weighting and classification, which we attribute to the higher accuracy of our model.

When compared to other deep learning approaches, our model outperformed those in [2], [5], [36] and [35] on Dataset-1 by 0.88%, 1.53%, 3.23% and 4.47%, respectively. On Dataset-2, our model outperformed the models in [37] and [38], which are based on convolutional and LSTM neural network architectures, by 3.25% and 2.09%. On Dataset-3, our model achieved a 2.23% higher accuracy rate than the deep learning approach presented in [6], which uses a genetic algorithm for feature selection and weighting, differing from our end-to-end approach of feature weighting and classifier training.

When comparing the recall of our model to current approaches in the literature, we see that our model performed better than all the other models on Dataset-2 and Dataset-3. Our model also achieved higher or equal results compared to all other models on Dataset-1, except for the model in [4], which achieved a slightly

Table 4.2: Results of alternative approaches.

Model	Accuracy	Precision	Recall	F_1-score	FPR
Dataset-1					
LSTM with weighting	0.95721	0.95329	0.94998	0.95163	0.03703
LSTM without weighting	0.97060	0.97428	0.95896	0.96656	0.02014
GRU with weighting	0.95360	0.95160	0.94324	0.94740	0.03817
GRU without weighting	0.96997	0.97228	0.95958	0.96589	0.02176
Dataset-2					
LSTM with weighting	0.94748	0.96111	0.94347	0.95220	0.04753
LSTM without weighting	0.96091	0.96820	0.96109	0.96463	0.03931
GRU with weighting	0.94870	0.96188	0.94493	0.95333	0.04662
GRU without weighting	0.96091	0.96613	0.96329	0.96471	0.04205
Dataset-3					
LSTM with weighting	0.91760	0.93469	0.91738	0.92595	0.08212
LSTM without weighting	0.92720	0.93957	0.93020	0.93486	0.07664
GRU with weighting	0.91520	0.92330	0.92593	0.92461	0.09854
GRU without weighting	0.92960	0.93983	0.93447	0.93714	0.07664

higher result (0.18%). However, our model obtained considerably higher accuracy than their model (0.37%). To determine if this difference in accuracy is statistically significant, we conducted a statistical test. Given that the data follows a binomial distribution(each sample is either classified correctly or misclassified), we performed Barnard's test with a significance level of 0.05. The null hypothesis was that our model does not improve accuracy, while the alternative hypothesis was that our model has higher accuracy. The test yielded a p value of 0.04355. Since this p-value is less than our chosen significance level, we have evidence to reject the null hypothesis in favor of the alternative, concluding that the difference in accuracy is statistically significant.

To compare the influence that different website features have on classification, we recorded the sigmoid activations of all the weighting parameters obtained after the model had completed its training phase. Since we evaluated our model using a 10-fold cross-validation technique, we calculated the average sigmoid activation of each weighting parameter across the 10 experiments. These values are provided in Table 4.4. The two features with the highest weighting parameters are the number of visitors and the number of pages they visit, and the use of SSL and trust in the certificate issuer. The lowest weight was assigned to the feature that checks if a host

Table 4.3: Comparison with other methods.

(a) Dataset-1

Method	Accuracy (%)	Recall (%)
Our approach with weighting	**97.53**	**96.65**
Our approach without weighting	**97.39**	**96.41**
Al-Sarem et al. [4]	97.16	96.83
Ali and Malebary [7]	96.83	95.27
Al-Ahmadi and Lasloum [2]	96.65	96.65
Vrbančič et al. [30]	96.65	N/A
Vrbančič et al. [29]	96.5	N/A
Lakshmi et al. [5]	96.0	N/A
Alqahtani [8]	95.20	N/A
Parra et al. [36]	94.3	93.67
Thabtah et al. [35]	93.06	91.12
Motlagh and Bardsiri [22]	93.42	92.27
Mohammad et al. [20]	92.48	N/A

(b) Dataset-2

Method	Accuracy (%)	Recall (%)
Our approach with weighting	**97.56**	**98.02**
Our approach without weighting	**97.15**	**97.28**
Al-Ahmadi and Lasloum [2]	95.73	95.73
Jalal and Naaz [39]	95.7	96.1
Wang et al. [38]	95.47	95.37
Al-Milli and Hammo [37]	94.31	N/A

(c) Dataset-3

Method	Accuracy (%)	Recall (%)
Our approach with weighting	**93.36**	**93.73**
Our approach without weighting	**92.80**	**92.74**
Khan et al. [33]	92.94	89.41
Kulkarni et al. [32]	91.50	90.97
Ali and Ahmed [6]	91.13	90.79
Almousa et al. [34]	88.67	N/A
Vrbančič et al. [30]	86.06	N/A

belongs to top phishing IPs or domains, based on PhishTank's and StopBadware's statistical reports. Comparing the six highest-ranked features using information gain and Chi-Square in [28], we find that they are the same as the six features with the highest weights in our approach.

Table 4.4: Average weighting parameter of each website feature.

Feature	Bounded Weight
Website Traffic	0.92
HTTPS (Hyper Text Transfer Protocol with Secure Sockets Layer)	0.91
Links in <Meta>, <Script> and <Link> tags	0.91
URL of Anchor	0.9
Subdomain and Multi Subdomains	0.9
Adding Prefix or Suffix Separated by (-) to the Domain	0.88
Server Form Handler (SFH)	0.84
Number of Links Pointing to Page	0.83
Google Index	0.82
Request URL	0.76
DNS Record	0.75
Using the IP Address	0.71
Age of Domain	0.7
Using Non-Standard Port	0.7
Submitting Information to Email	0.67
Domain Registration Length	0.65
The Existence of “HTTPS” Token in the Domain Part of the URL	0.65
PageRank	0.65
Website Forwarding	0.63
Redirecting using “//”	0.53
Using URL Shortening Services “TinyURL”	0.5
Using Pop-up Window	0.47
Long URL to Hide the Suspicious Part	0.42
Abnormal URL	0.41
Favicon	0.38
URL’s having “@” Symbol	0.38
IFrame Redirection	0.31
Disabling Right Click	0.29
Status Bar Customization	0.21
Statistical-Reports Based Feature	0.08

5. Conclusion and future work

In this paper, we addressed the problem of phishing website detection by utilizing discrete descriptive website features. We first designed an embedding layer to learn

vectorized representations of these features. Inspired by the CancelOut layer, we developed an embedding weighting layer to model the varying influence of different features on phishing detection. By incorporating these two layers, we proposed a convolutional neural network-based classifier and conducted an extensive evaluation to confirm its efficacy.

An advantage of using discrete descriptive features is their interpretability. To the best of our knowledge, we are the first to learn embeddings of these website features. These embeddings enabled us to apply more powerful neural network architectures, such as convolutional neural networks. Combined with descriptive features, it also helps us to have a better understanding of how the system works, which is more challenging with other commonly used types of embeddings, like character or word embeddings.

In the future, collecting a larger dataset of phishing and legitimate websites and extracting descriptive features from them could be of great importance to the field. As convolutional neural networks typically perform better with larger datasets, we expect that our model would be able to learn effectively from such datasets.

Another direction for further improvement would be incremental learning from phishing websites. Over time, new types of phishing websites will emerge, and it is crucial to adapt models to these changes. Instead of retraining the entire model with both old and new samples, it would be advantageous if we could update the model with only the new samples, without needing to store all the old ones.

As the number of phishing attacks continues to rise, we believe that the field requires more attention. Contrary to the commonly used fully connected feed-forward neural network models and various two-step approaches, we explored an end-to-end convolutional neural network approach. We also demonstrated the efficacy of website feature embedding and weighting for phishing website detection and anticipate seeing more similar approaches in the future.

REFERENCES

1. E. A. G. Abad, J. R. A. Ferrer and P. C. Naval Jr.: *Phishing Website Classification Using Features of Web Addresses and Web Pages*. In: Proc. 20th Philippine Computing Science Conference, Baguio City, Philippines, 2020.
2. S. Al-Ahmadi and T. Lasloum: *PDMLP: Phishing Detection Using Multilayer Perceptron*. International Journal of Network Security & Its Applications, **12**(3) (2020), 59–72.
3. Anti-Phishing Working Group: *Phishing activity trends report, 1st quarter 2021*. `https://docs.apwg.org/reports/apwg_trends_report_q1_2021.pdf`.
4. M. Al-Sarem, F. Saeed, Z. G. Al-Mekhlafi, B. A. Mohammed, T. Al-Hadhrami, M. T. Alshammari, A. Alreshidi and T. S. Alshammari: *An Optimized Stacking Ensemble Model for Phishing Websites Detection*. Electronics, **10**(11) (2021).

5. L. Lakshmi, M. P. Reddy, C. Santhaiah and U. J. Reddy: *Smart Phishing Detection in Web Pages using Supervised Deep Learning Classification and Optimization Technique ADAM*. Wireless Personal Communications, **118**(4) (2021), 3549–3564.
6. W. Ali and A. A. Ahmed: *Hybrid intelligent phishing website prediction using deep neural networks with genetic algorithm-based feature selection and weighting*. IET Information Security, **13**(6) (2019), 659–669.
7. W. Ali and S. Malebary: *Particle Swarm Optimization-Based Feature Weighting for Improving Intelligent Phishing Website Detection*. IEEE Access, **8** (2020), 116766–116780, `https://doi.org/10.1109/access.2020.3003569`.
8. M. Alqahtani: *Phishing Websites Classification Using Association Classification (PWCAC)*. In: 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, `https://doi.org/10.1109/iccisci.2019.8716444`.
9. V. Borisov, J. Haug and G. Kasneci: *Cancelout: A layer for feature selection in deep neural networks*. In: International Conference on Artificial Neural Networks, 2019, 72–83.
10. I. Fister Jr., D. Fister and X. S. Yang: *A hybrid bat algorithm*. Elektrotehniški Vestnik, **80**(1-2) (2013), 1–7.
11. Google LLC: *Google colab*. 2017, `https://colab.research.google.com/`
12. W. Hadi, F. Aburub and S. Alhawari: *A New Fast Associative Classification Algorithm for Detecting Phishing Websites*. Applied Soft Computing, **48** (2016), 729–734.
13. R. Islam and J. Abawajy: *A Multi-Tier Phishing Detection and Filtering Approach*. Journal of Network and Computer Applications, **36**(1) (2013), 324–335.
14. D. P. Kingma and J. Ba: *Adam: A method for stochastic optimization*. In: 3rd International Conference on Learning Representations, San Diego, CA, USA, 2015.
15. T. Mikolov, K. Chen, G. S. Corrado and J. Dean: *Efficient Estimation of Word Representations in Vector Space*. In: ICLR (Workshop Poster), 2013.
16. R. M. Mohammad, L. McCluskey and F. Thabtah: *UCI Machine Learning Repository - Phishing Websites Dataset*. Irvine, University of California, School of Information and Computer Science, 2012, `https://archive.ics.uci.edu/ml/datasets/Phishing+Websites`.
17. N. Abdelhamid: *Irvine, CA: University of California, School of Information and Computer Science, Machine Learning Repository*. 2016, `https://archive.ics.uci.edu/dataset/379/website+phishing`.
18. N. Abdelhamid, A. Ayesh and F. Thabtah: *Phishing detection based associative classification data mining*. Expert Systems with Applications, **41**(13) (2014), 5948–5959.
19. R. M. Mohammad, F. Thabtah and L. McCluskey: *An Assessment of Features Related to Phishing Websites Using an Automated Technique*. In: International Conference for Internet Technology and Secured Transactions, London, UK, 2012, 492–497.
20. R. M. Mohammad, F. Thabtah and L. McCluskey: *Predicting phishing websites based on self-structuring neural network*. Neural Computing and Applications, **25**(2) (2014), 443–458.
21. N. Moradpoor, B. Clavie and B. Buchanan: *Employing Machine Learning Techniques for Detection and Classification of Phishing Emails*. In: 2017 Computing Conference, London, UK, 2017, 149–156.

22. F. Parandeh Motlagh and A. Khatibi Bardsiri: *Detecting fake websites using swarm intelligence mechanism in human learning*. International Journal of Engineering, **31**(10) (2018), 1642–1650, https://doi.org/10.5829/ije.2018.31.10a.05.

23. S. Hochreiter and J. Schmidhuber: *Long short-term memory*. Neural computation, **9**(8) (1997), 1735–1780.

24. J. Chung, C. Gulcehre, K. Cho and Y. Bengio: *Empirical evaluation of gated recurrent neural networks on sequence modeling*. 2014, arXiv preprint arXiv:1412.3555.

25. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer: *Automatic differentiation in pytorch*. In: 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017.

26. R. K. V. Penmatsa and P. Kakarlapudi: *Web Phishing Detection: Feature Selection Using Rough Sets and Ant Colony Optimisation*. International Journal of Intelligent Systems Design and Computing, **2**(2) (2018), 102–113, https://doi.org/10.1504/ijisdc.2018.096329.

27. M. Rajab: *Visualisation Model Based on Phishing Features*. Journal of Information & Knowledge Management, **18**(01) (2019), https://doi.org/10.1142/s0219649219500102.

28. F. Thabtah and N. Abdelhamid: *Deriving Correlated Sets of Website Features for Phishing Detection: A Computational Intelligence Approach*. Journal of Information & Knowledge Management, **15**(04) (2016), https://doi.org/10.1142/s0219649216500428.

29. G. Vrbančič, I. Fister Jr. and V. Podgorelec: *Swarm intelligence approaches for parameter setting of deep learning neural network: Case study on phishing websites classification*. In: Proceedings of the 8th international conference on web intelligence, mining and semantics, New York, United States, 2018, 1–8.

30. G. Vrbančič, I. Fister Jr. and V. Podgorelec: *Parameter setting for deep neural networks using swarm intelligence on phishing websites classification*. International Journal on Artificial Intelligence Tools, **28**(06) (2019).

31. X. S. Yang: *A new metaheuristic bat-inspired algorithm*. In: Nature inspired cooperative strategies for optimization, Granada, Spain, 2010, 65–74.

32. A. D. Kulkarni and L.Ł. Brown III: *Phishing website detection using machine learning*. In: Computer Science Faculty Publications and Presentations, Paper 20, 2019, http://hdl.handle.net/10950/1862.

33. S. A. Khan, W. Khan and A. Hussain: *Phishing attacks and websites classification using machine learning and multiple datasets (a comparative analysis)*. In: Intelligent Computing Methodologies: 16th International Conference, Bari, Italy, 2020, 301–313.

34. M. Almousa, T. Zhang, A. Sarrafzadeh and M. Anwar: *Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?*. Security and Privacy, **5**(6) (2022), https://doi.org/10.1002/spy2.256.

35. F. Thabtah, R. M. Mohammad and L. McCluskey: *A dynamic self-structuring neural network model to combat phishing*. In: International joint conference on neural networks (IJCNN), Vancouver, Canada, 2016, 4221–4226.

36. G. D. L. T. Parra, P. Rad, K. K. R. Choo and N. Beebe: *Detecting Internet of Things attacks using distributed deep learning*. Journal of Network and Computer Applications, **163** (2020), https://doi.org/10.1016/j.jnca.2020.102662.

37. N. Al-Milli and B. H. Hammo: *A convolutional neural network model to detect illegitimate URLs*. In: 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 2020, 220–225.

38. S. Wang, S. Khan, C. Xu, S. Nazir and A. Hafeez: *Deep learning-based efficient model development for phishing detection using random forest and BLSTM classifiers*. Complexity, **1** (2020), https://doi.org/10.1155/2020/8694796.

39. K. Jalal and S. Naaz: *Detection of phishing websites using machine learning approach*. In: International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM), Jaipur, Rajasthan, India, 2019.